الدرس الخامس

أهلا بكم في الدرس الخامس من سلسلة دروس تعلم ال Xna , سوف نقوم في هذا الدرس برسم المدفع و عمل التدوير "Rotation" عليه, إن شاء الله.

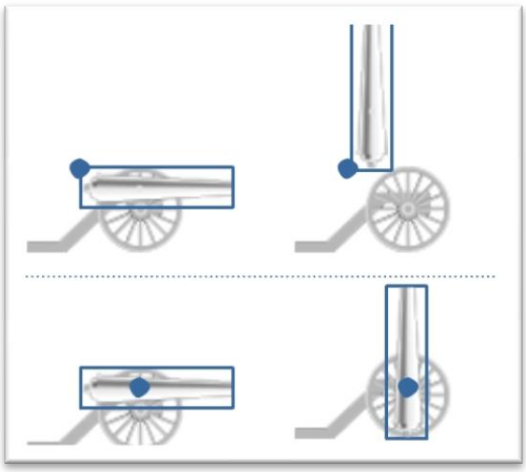
لقد قمنا حتى الآن تقريبا بتغطية غالبية المفاهيم حول الرسم الثنائي الأبعاد , إلا مفهوم واحد ألا وهو التدوير. في هذا الدرس سوف نقوم برسم المدفع على ظهر حاملة المدفع التي رسمناها في الدرس السابق, بناء على زاوية التدوير التي سوف نخزنها في المصفوفة PlayerData.

من المفروض أنك قد قمت فعليا بإستيراد صورة المدفع و قمت بربطها مع المتغير cannonTexture قبل درسين. دعنا الآن نناقش كيف سنقوم بتدوير هذا المدفع.

كما ذكر في الدرس السابق فإن ال Xna تعتبر أن نقطة الأصل للصورة ثنائية الأبعاد تمثل الركن العلوي الأيسر. هذا مهم بالنسبة لنا لسببين, حيث قمنا بتغطية السبب الأول في الدرس السابق:

- تقوم ال Xna برسم الصورة على الشاشة بحيث تكون نقطة الأصل لها فوق النقطة التي نقوم بإرسالها إلى الدالة SpriteBatch.Draw().
- السبب الثاني , أنك عندما تقوم بتدوير الصورة, تقوم ال Xna بتدويرها حول نقطة الأصل الخاصة بها. و هذه نتيجة طبيعية مربتطة مباشرة بالسبب الأول بحيث: أنه و بهذه الطريقة: نقطة الأصل تبقى نفسها في الموقع الذي تقوم بتحديده, بغض النظر عن طبيعة التدوير .

الصورة التالية توضح ذلك:



في الجزء العلوي, تم رسم المدفع بإعتبار أن نقطة الأصل له هي الركن الاعلى الأيسر. الآن عندما نريد تطبيق التدوير على الصورة كما الحال في الصورة العلويه على اليمين, سيتم تدوير المدفع حول نقطة الأصل الخاصة به. تخيل لو أننا قمنا بتدوير المدفع 90 درجه اخرى, سيظهر المدفع و كأنه خارج منطقة الحمالة !
ما نحتاجه هو تحديد نقطة الأصل في مركز التدوير, بحيث يكون موقع هذه النقطة ثابت أثناء عملية التدوير. هذه النقطة تمثل النقطة الزرقاء في الجزء الأسفل من الرسم السابق, الآن عندما يتم تدوير المدفع بزاوية 90 درجة, سوف تكون النتيجة أفضل بكثير, كما يظهر في القسم الأسفل على اليمين من الصورة.

إذن كيف يمكننا عمل ذلك في الXna؟ بسهولة جدا ☺ . قم بإيجاد السطر في الكود الذي يقوم برسم حاملة المدفع (موجود في الدالة (DrawPlayers() , وقم بإضافة الكود التالي مباشرة قبل ذلك السطر:

```
 int xPos = (int)player.Position.X;
 int yPos = (int)player.Position.Y;
 Vector2 cannonOrigin = new Vector2(11, 50);

 spriteBatch.Draw(cannonTexture, new Vector2(xPos + 20, yPos - 10), null,
player.Color, player.Angle, cannonOrigin, playerScaling, SpriteEffects.None,
1);
```

نحن بحاجة إلى تحديد موقعين:

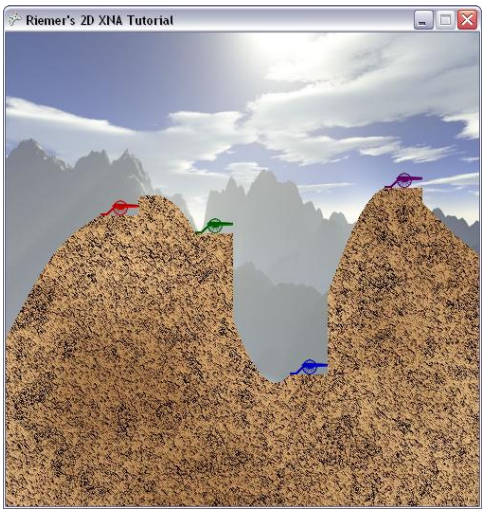
- نقطة الأصل الجديدة الخاصة بصورة المدفع
- الموقع على الشاشة الذي يعبر عن موقع رسم النقطة السابقة.

حجم صورة المدفع هو 70 * 20 بكسل, لهذا قمت بإختيار الإحداثي (11,50) كنقطة الأصل. حاول فتح الصورة و الوصول إلى هذه النقطة سوف تجد أنها موجوده على المركز التدويري للصورة (مقبض التدوير). يجب أن تلاحظ أن هذا الموقع يجب تحديده على الصورة بحجمها الأصلي.

بعدها يلزمنا تحديد أين على الشاشة سوف يتم رسم هذه النقطة. حيث نريد أن يتم ذلك على بعد 20 بكسل من اليمين و 10 بكسل للأعلى, بالنسبة إلى موقع اللاعب, الذي يمثل الركن الأعلى الأيسر من صورة حامل المدفع.

أخيرا, قمنا بوضع زاوية الدوران بدلا من الصفر. يمكنك أن تجرب بعض القيم بنفسك, بإستخدام الدالة MathHelper.ToRadians(). في السطر الأخير قمنا بأخذ قيمة الدوران و اللون المخزنان في مصفوفة ال PlayerData .

الآن دعنا نجرب تشغيل الكود, سترى أن كل المدافع موجهة إلى اليمين كما في الصورة, أي أنها مدورة بزاوية 90 درجة كما فعلنا سابقا.



في الدرس القادم سوف نتعلم كيف نقرأ المدخلات من لوحة المفاتيح, بحيث يمكننا تعديل الزاويه أثناء تنفيذ البرنامج!

كود المشروع حتى الآن:

```
using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

namespace XNAtutorial
{
    public struct PlayerData
    {
        public Vector2 Position;
        public bool IsAlive;
        public Color Color;
        public float Angle;
        public float Power;
    }

    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        GraphicsDevice device;

        int screenWidth;
        int screenHeight;

        Texture2D backgroundTexture;
        Texture2D foregroundTexture;
        Texture2D carriageTexture;
        Texture2D cannonTexture;

        PlayerData[] players;
        int numberOfPlayers = 4;
        float playerScaling;

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        protected override void Initialize()
        {
            graphics.PreferredBackBufferWidth = 500;
            graphics.PreferredBackBufferHeight = 500;
            graphics.IsFullScreen = false;
            graphics.ApplyChanges();
            Window.Title = "Riemer's 2D XNA Tutorial";

            base.Initialize();
        }

        protected override void LoadContent()
        {
            device = graphics.GraphicsDevice;
            spriteBatch = new SpriteBatch(device);

            screenWidth = device.PresentationParameters.BackBufferWidth;
            screenHeight = device.PresentationParameters.BackBufferHeight;


           backgroundTexture = Content.Load<Texture2D> ("background");
           foregroundTexture = Content.Load<Texture2D> ("foreground");
           carriageTexture = Content.Load<Texture2D> ("carriage");
           cannonTexture = Content.Load<Texture2D> ("cannon");
           SetUpPlayers();
           playerScaling = 40.0f / (float)carriageTexture.Width;
       }

       private void SetUpPlayers()
       {
           Color[] playerColors = new Color[10];
           playerColors[0] = Color.Red;
```

```
            playerColors[1] = Color.Green;
            playerColors[2] = Color.Blue;
            playerColors[3] = Color.Purple;
            playerColors[4] = Color.Orange;
            playerColors[5] = Color.Indigo;
            playerColors[6] = Color.Yellow;
            playerColors[7] = Color.SaddleBrown;
            playerColors[8] = Color.Tomato;
            playerColors[9] = Color.Turquoise;

            players = new PlayerData[numberOfPlayers];
            for (int i = 0; i < numberOfPlayers; i++)
            {
                players[i].IsAlive = true;
                players[i].Color = playerColors[i];
                players[i].Angle = MathHelper.ToRadians(90);
                players[i].Power = 100;
            }

            players[0].Position = new Vector2(100, 193);
            players[1].Position = new Vector2(200, 212);
            players[2].Position = new Vector2(300, 361);
            players[3].Position = new Vector2(400, 164);
        }

        protected override void UnloadContent()
        {
        }

        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            base.Update(gameTime);
        }

        protected override void Draw(GameTime gameTime)
        {
            graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

            spriteBatch.Begin();
            DrawScenery();
            DrawPlayers();
            spriteBatch.End();

            base.Draw(gameTime);
        }

        private void DrawScenery()
        {
            Rectangle screenRectangle = new Rectangle(0, 0, screenWidth, screenHeight);
            spriteBatch.Draw(backgroundTexture, screenRectangle, Color.White);
            spriteBatch.Draw(foregroundTexture, screenRectangle, Color.White);
        }

        private void DrawPlayers()
        {
            foreach (PlayerData player in players)
            {
                if (player.IsAlive)
                {

                    int xPos = (int)player.Position.X;
                    int yPos = (int)player.Position.Y;
                    Vector2 cannonOrigin = new Vector2(11, 50);

                    spriteBatch.Draw(cannonTexture, new Vector2(xPos + 20, yPos - 10), null,
player.Color, player.Angle, cannonOrigin, playerScaling, SpriteEffects.None, 1);
                    spriteBatch.Draw(carriageTexture, player.Position, null, player.Color, 0, new
Vector2(0, carriageTexture.Height), playerScaling, SpriteEffects.None, 0);
                }
            }
        }
    }
}
```